Scheduling Reclaimer Operations in the Stockyard to Minimize Makespan

Wang, Chao; Lu, Xi wen; Sitters, René

published in
Acta Mathematicae Applicatae Sinica
2018

DOI (link to publisher)
10.1007/s10255-018-0758-6

document version
Publisher's PDF, also known as Version of record

Acta Mathematicae Applicatae Sinica, English Series
Vol. 34, No. 3 (2018) 597–609
DOI: 10.1007/s10255-018-0758-6
http://www.ApplMath.com.cn & www.SpringerLink.com

Acta Mathematicae Applicatae Sinica, English Series

Scheduling Reclaimer Operations in the Stockyard to Minimize Makespan

Chao WANG[1], Xi-wen LU[1,†], René SITTERS[2]

[1]East China University of Science and Technology, 200237, Shanghai, China ([†]E-mail: xwlu@ecust.edu.cn)
[2]VU University Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Abstract This paper considers a reclaimer scheduling problem in which one has to collect bulk material from stockpiles in the quay in such a way that the time used is minimized. When reclaimers are allowed to work on the same stockpile simultaneously, a fully polynomial time approximation scheme (FPTAS) is designed. Further, we present a 2-approximation algorithm in the case that any stockpile can be handled by only one reclaimer at a time. When the number of reclaimers is two, we give a 3/2-approximation algorithm. Numerical experiments show that the algorithms perform much better than our worst case analysis guarantees.

Keywords scheduling; approximation algorithm; performance ratio; numerical simulation
2000 MR Subject Classification 90B35; 90C59

1 Introduction

The exportation of the bulk material is very important to the economic growth of a country. For example, the number of coal exports of China is up to 5.33 million tons in 2015, making a profit of 498.75 million USD, according to NBSC[16]. In order to stand out from the competition, the export corporation needs to satisfy their customers with high efficiency and products quality. Here, the bulk material stockyard plays an important role in the exportation business by serving as an interface between inland and sea transports. The bulk material is transported to the quay beforehand, and stockpiled at the stockyard of the quay before it is loaded onto the ship, so that the bulk material on the customer's request list can be collected directly from the stockyard in less time when the vessel of the customer arrives. Moreover, the customers may have more detailed requests. For example, the customer who requires iron ore will set strict limits on the mixture of iron, silica, alumina and calcium oxide. Therefore, stockpiles with different compositions are built to meet customer's various requests. The stockyard can act as buffers and the variation in the composition (namely, grade) of the certain bulk material can be reduced due to the stockpiles.

The most important objective for the planner of the corporation is to increase the throughput of the quay. When there is an order, the customer will inform the export corporation of the detailed request list. After the export corporation gets the requests, the amount of the bulk material to be reclaimed from each stockpile is determined. Usually several reclaimers are employed to finish the customer's order. The most common machine which is used for reclaiming the bulk material is Bucket Wheel Reclaimer (BWR). The cost of operating the reclaimers is higher than the other operations like delivering the bulk material from the reclaimers to the ship. Besides, an effective schedule of the reclaimers will definitely increase the efficiency of the reclaiming operation. As a result, it is meaningful to get a better understanding of the reclaimer scheduling problem in order to save time, increase the throughput and make profits.

Manuscript received September 13, 2017. Revised February 26, 2018.
Supported by the National Natural Science Foundation of China (No. 11371137 and No.71431004).

The stockpiles are located on parallel strips and each stockpile has a given location and a given volume, which is more than required. The grades of the stockpiles are different from each other, so each stockpile represents a unique grade. There is a straight track lying between every two consecutive strips. If the stockyard has $m + 1$ parallel strips, then there are m tracks and each BWR is located on a single rail track, thus they can only move (back and forth) along the given straight path. Each BWR can handle the stockpiles which are located on the two strips on both sides of the BWR.

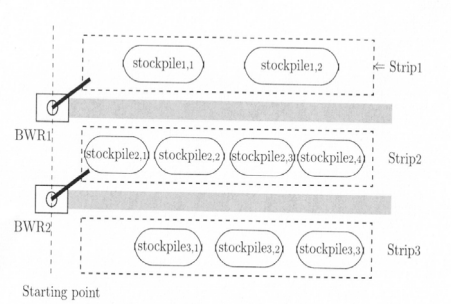

Figure 1. An Example Overview of the Stockyard

In the example of Figure 1, there are three lines of strips and two BWR machines. BWR_1 can process all the six stockpiles on Strip 1 and 2, while BWR_2 can handle the seven stockpiles on Strip 2 and 3. We assume that all BWRs keep a uniform speed when moving along the track. Furthermore, the reclaiming speed of all machines is also a constant. The BWR has to move to the location of the stockpile and then begins to reclaim the amount of bulk material the customer needs. In most cases, it is not allowed that two BWRs work at the same stockpile at the same time, because the two reclaimers may collide with each other in this way. However, if two BWRs are able to work at the same stockpile at the same time, a better solution may be obtained.

The scheduling of bulk material share some similarities with scheduling of containers in the stockyard. In the latter problem, container carriers are employed to deliver the containers from the stockyard to the vessel. The primary goal is to minimize the time it takes to complete the delivery of containers. The two problems also have many differences. The containers have fixed sizes, so the number of the processing time of loading/unloading containers is limited. However, the size of bulk material stockpiles is arbitrary. The BWR can only travel along a straight track, but the movement of container carriers may be more flexible.

Kim and Kim[8] dealt with a routing problem of a single container carrier in the stockyard to minimize the total travel time, and they formulated a mixed integer programming to solve the optimal route as well as the optimal number of containers to be picked up each time. Later, two heuristic algorithms were introduced by Kim and Kim[9] to reduce excessive computational time of the same problem. Yard cranes are also commonly used for handling containers, but they are less mobile. Ng and Mak[18] studied the problem of scheduling yard cranes to minimize total waiting time with different job ready times. A scheduling heuristic was developed to solve this problem.

As for multiple container carriers, Zhang, et al.[21] considered the problem to minimize total delayed workload with a set of yard cranes. A mixed integer programming model was formulated to find the times and routes of crane movements which was solved by Lagrangean relaxation. Ng[17] generalized Ng and Mak's[18] result to address the problem of multiple yard cranes to minimize total waiting times with different ready times. The problem was formulated as an integer program, but solved by a dynamic programming-based heuristic. Daganzo[4] studied a

crane scheduling problem in which multiple cranes work on the holds of the ships. The ships arrive at different times, and they cannot depart until all the holds are finished. The objective is to minimize the sum of departure delays of the ships. A mixed integer programming and some heuristics were presented.

Recently, scheduling of bulk material in the stockyard becomes popular. Robenek, et al.[19] presented an integrated model of berth allocation and yard assignment problems in bulk material ports, where a branch-and-price algorithm was designed to obtain exact solutions for small instances and a heuristic critical-shaking neighborhood search to find solutions quickly for large instances. Menezes, et al.[14,15] considered an integrated production planning and scheduling problem in bulk cargo to define the amounts and the routes of products from the supply nodes to the terminal. Hu and Yao[7] discussed a reclaimer scheduling problem in the bulk material terminal. Multiple reclaimers are dispatched to minimize makepan. Each task has a sequence dependent setup time and should be assigned to an eligible reclaimer. This problem was formulated as a mixed integer programming and a genetic algorithm was proposed to solve this problem. Sun and Tang[20] investigated a reclaimer scheduling problem at an iron ore terminals. Each operation has a release time and due date, and the objective is to minimize total weighted tardiness. They proposed a mixed integer programming and solved by an improved Bender decomposition. Angelelli, et al.[2] studied reclaiming bulk material in the stockyard as well. Every stockpile is close to only one track in their paper and there is one or two reclaimers on the track, which implies that the stockpiles can only be reclaimed by one of the reclaimers on that track. They proved NP-hardness of different variants and provided polynomial algorithms for some variants.

A large amount of study on the technical aspect of operating the BWR has been done in recent years. Lu[12] considered the kinematics and dynamics of a typical BWR and modeled it as a robot arm. Zhao, et al.[22] used the laser scanner sensors to perceive the grade of stockpiles. Moreover, Lu and Myo[13] also divided a stockpile into voxels to have a better control of the quality grade. Similarly, in the work by Lee, et al.[10], the reclaimers are equipped with 3D range finder to measure the shape of the stockpile for optimizing the moving paths of the reclaimers. A better localization technique was raised by Zhao, et al.[23] using the UKF algorithm to fuse DGPS and encoder data.

Motivated by the practical reclaimer operations, we consider two problems in the paper. The first problem is that the reclaimers are allowed to work on the same stockpile at the same time, and the other problem is that the reclaimers are not allowed to work on the same stockpiles simultaneously. For the first problem, we design an FPTAS and for the second problem we give a 2-approximation algorithm for multiple machines and a 3/2-approximation for the case of two machines. In the end, numerical experiments are given to evaluate the effect of the approximation algorithms.

The remainder of this paper is organized as follows. In Section 2, the problem is formulated as a combinatorial optimization problem and some assumptions and notations are introduced. In Section 3, the reclaimers are allowed to work on the same stockpile simultaneously, and in Section 4, the reclaimers are not. In Section 5 some computational results are shown. Concluding remarks are given in Section 6.

2 Mathematical Modeling and Formulation

We will study the problem based on the knowledge of scheduling theory, which is an important part of combinatorial optimization. The following assumptions, which will be used in this paper, match the real-life situation.

- There is exactly one reclaimer on each track, and the reclaimer can handle the stockpiles

lying on both sides of the track.

- All reclaimers are identical. It means they have the same reclaiming speed, which is a constant and all reclaimers have the same traveling speed, w.l.o.g. a constant speed of 1. In the beginning, every reclaimer is located at the left end, i.e. the starting point, as seen in Figure 1.

- The grade of each stockpile is different from the others, and the total volume of each stockpile is more than required.

- There may be more than one request on the same stockpile and these are handled independently by the reclaimers.

Then we formulate this problem as a scheduling problem. Here we view each BWR as one machine, each stockpile as a location and each independent request as a job. The objective function is to minimize the makespan. Each job can be processed by one of at most two machines, and each job has a processing time and a job location. Then it is related to a parallel machine scheduling problem with the job assignment restrictions to minimize the makespan. Since we know the amounts of bulk material to be reclaimed at each stockpile in advance, moreover, the machines are identical and the reclaiming speed is known, the processing time of each job can be determined by the amount of the corresponding request. A machine has to travel to the location of the job, which is known beforehand, and then begins to process that job. The processing of the jobs cannot be preempted. Each machine can only work on one job at a time. The load of the machine is the total travel time plus the total job processing time. The makespan is the maximal load among all the machines.

The classical parallel machine scheduling problem, denoted by $P||C_{\max}$ is known to be NP-hard in the strong sense[5], and there is a PTAS (polynomial time approximation scheme) for the problem[1]. When each job can only be scheduled by one of at most two machines, the problem is called *Graph Balancing* problem and a 1.75-approximation algorithm was presented by Ebenlendr, et al.[3]. Further, when the graph is restricted to a tree, there is an FPTAS (fully polynomial time approximation scheme) for this restricted graph balancing problem[11]. The problem in this paper is similar to the graph balancing problem, but every job has a location in this problem. Moreover, the machines may not be allowed working on the same location at the same time, which makes the problem more complicated.

At first, the strips are indexed by the order similar to Figure 1. We assume that there are m machines and $m+1$ strips, and machine i is located on the track between the i-th and the $(i+1)$-st strip. In the i-th strip, there are n_i jobs. The number of jobs in total is $n = \sum_{i=1}^{m+1} n_i$. We sort the jobs on every strip in the increasing order of the job locations, and ties are broken arbitrarily. The job location is the distance from the starting point to the position of the job on the corresponding strip, which also means the time from the starting point to the position of the job on that strip, due to the assumption that all the machines' traveling speed is 1. Now we give some notations:

M_i: i-th machine $(i = 1 \cdots m)$.
$J_j^{(i)}$: the j-th job on strip i $(j = 1 \cdots n_i)$.
$p_j^{(i)}$: the processing time of $J_j^{(i)}$ $(j = 1 \cdots n_i)$.
$s_j^{(i)}$: the location of job $J_j^{(i)}$ $(j = 1 \cdots n_i)$.

When i is fixed and no ambiguity arises we sometimes write j, p_j and s_j in stead of, respectively, $J_j^{(i)}, p_j^{(i)}$ and $s_j^{(i)}$.

3 Simultaneous Handling of Stockpiles

In this section, the machines can process the jobs at the same stockpile simultaneously, which means that two machines are able to work on different jobs located on the same stockpile at the same time and each job can only be handled by one machine. First of all, it is easy to find out that the problem is NP-hard. In fact, when there are only two machines and all jobs are located at the same stockpile on the middle strip, it becomes *load balancing* which is a classical NP-complete problem. Here we provide an FPTAS for this case.

Lemma 1. *There exists an optimal solution of this problem, in which all the machines travel from the left side (the starting point) to the right side, without turning around.*

Proof. Suppose to the contrary that there is an optimal solution in which a machine does not follow the direct path from left to right. So the machine must work on a job j_1 prior to job j_2 with job location $s_{j_1} > s_{j_2}$. It implies that when the machine reaches job j_1, it has definitely passed job j_2. We exchange the processing sequence of job j_1 and j_2, keeping the other job sequence unchanged to get a new schedule. Thus job j_2 is scheduled prior to job j_1 in the new schedule and the objective value does not increase. By repeating the procedure above, we will get an optimal schedule in which all the machines travel from left to right. □

First, we design a dynamic programming which gets the optimal solution in pseudo-polynomial time. For each $i \in \{2 \cdots, m\}$, define the set S_i as all triples (p, k, l), each of which corresponds to a feasible partition of strip i in the sense that the last job of strip i done by machine M_{i-1} is l, the last job of strip i done by machine M_i is k, and the total processing time done by M_i on strip i is exactly p. That means $(p, k, l) \in S_i$ if and only if the following conditions hold:

- $k, l \in \{0, \cdots, n_i\}$, $k \neq l$, and $n_i \in \{k, l\}$
- If $l = n_i$ then there exists a subset of jobs from $1, \cdots, k$ with total processing time p.
- If $k = n_i$ then there exists a subset of jobs from $1, \cdots, l-1$ with total processing time $p - \sum_{r=l+1}^{n_i} p_r$.

Here, $k = 0$ ($l = 0$) means that all jobs on strip i are processed by M_{i-1} (M_i). For each $i = 2, \cdots, m$ and $(p, k, l) \in S_i$ denote by $F_i(p, k, l)$ the optimal makespan for the instance restricted to machines $M_i, M_{i+1}, \ldots, M_m$ and strips $i, i+1, \ldots, m+1$ and under the restriction that

- the total processing time done by M_i on jobs of strip i is exactly p
- the last job on strip i processed by M_i is k

We can check whether a triple $(p, k, l) \in S_i$ in $O(nP)$ time since this is a simple *subset sum problem*[6]. Here P is the total processing time of all the jobs. There are $O(nP)$ triples in total so the overall running time for computing all sets S_i is $O(mn^2P^2)$. Denote by P_i the total processing time of the jobs on strip i. The values $F_i(p, k, l)$ are obtained by the following dynamic programming.

Initialization:

$$F_m(p, k, l) = P_{m+1} + p + \max\{s_k^{(m)}, s_{n_{m+1}}^{(m+1)}\}, \qquad \text{for all } (p, k, l) \in S_m.$$

Recurrence relations:

$$F_i(p, k, l) = \min_{(\tilde{p}, \tilde{k}, \tilde{l}) \in S_{i+1}} \max\{F_{i+1}(\tilde{p}, \tilde{k}, \tilde{l}), P_{i+1} - \tilde{p} + p + \max\{s_k^{(i)}, s_{\tilde{l}}^{(i+1)}\}\}.$$

The optimal value is $F_1(P_1, n_1, 0)$.

The dynamic programming has $O(mnP)$ states, and the time complexity of getting the value of each state is $O(nP)$. So the total time complexity of the dynamic programming is $O(mn^2P^2)$. If we store for each triple $(p, k, l) \in S_i$ a corresponding partition of the jobs of strip i then the DP also provides an optimal assignment of jobs to machines. By Lemma 1, it is optimal to schedule the jobs by increasing order of the locations on each machine.

Next, we will construct an FPTAS. Given an instance I we first turn it into an instance I' by rounding the processing times. Then we use the above dynamic programming to find an optimal assignment for I' and use that as a solution for I. A lower bound L on the optimal value is $L = \max\{p_{\max}, P/m\}$, where $p_{\max}$ is the maximal value of all the job processing times, and P is the total processing times of all the jobs. Let $\delta = \varepsilon L/n$, where ε is a small positive number. Now we get the new instance I' by rounding the processing time of every job in instance I down to $p_j'^{(i)} = \lfloor p_j^{(i)}/\delta \rfloor$. Note that $\delta\lfloor p_j^{(i)}/\delta \rfloor \le p_j$. Hence, $\delta \mathrm{OPT}(I') \le \mathrm{OPT}(I)$, where $\mathrm{OPT}(I)$ and $\mathrm{OPT}(I')$ are the optimal value of the instance I and I', respectively. Note that some jobs in I' may have processing time of 0 after rounding, but we can not remove them from the instance, since these jobs still have nonzero job locations. Therefore, the number of jobs in instance I is equal to the number in instance I'. For instance I', we use the dynamic programming to get the optimal solution.

Algorithm H1.

Step 1. *Round the processing times and apply the DP to the rounded instance to compute an assignment of jobs to machines.*

Step 2. *Assign jobs to machines as computed in Step 1 and for each machine, schedule the jobs in increasing order of job locations.*

When the instance I' is rounded back to I, the set of jobs on each machine is kept the same, and the rounding error is only caused by the change of job processing time. For each job j on strip i, $p_j^{(i)} \le \lfloor p_j^{(i)}/\delta \rfloor \delta + \delta$ and there are at most n jobs done by each machine. Hence, the objective value obtained by the algorithm is at most

$$\delta \mathrm{OPT}(I') + n\delta \le \mathrm{OPT}(I) + \varepsilon L \le (1+\varepsilon)\,\mathrm{OPT}(I).$$

Further, $P' \le P/\delta \le mL/\delta = mn/\varepsilon$. The overall time complexity is at most $O(n^4m^3/\varepsilon^2)$.

Theorem 2. *Algorithm H1 gives an FPTAS for the reclaimer scheduling problem when machines can work simultaneously on the same stockpile.*

4 One at a Time Handling of Stockpiles

Two machines which are handling the same stockpile simultaneously may collide with each other. Therefore, in most reclaimer systems, a stockpile can only be handled by one reclaimer at a time. This more realistic model will be considered in this section. Note that Lemma 1 of Section 3 does not hold any more in this case, i.e., machines may have to move back and forth in an optimal schedule. For example, consider instance $\widehat{I}$ with two machines and three strips of jobs. J_1 and J_2 are on the second strip, J_3 on the first strip and J_4 on the third strip. $p_1 = p_2 = p$, $s_1 = s_2 = \varepsilon$. $p_3 = p - 2\varepsilon$, $s_3 = 2\varepsilon$, $p_4 = \varepsilon$ and $s_4 = p - \varepsilon$. The optimal solution of this instance is that M_1 firstly processes J_3 and then travels back to process J_1, and M_2 handles J_2 and then travels to finish J_4. The optimal value is $2p + \varepsilon$.

This possibility of moving backwards makes the problem more difficult than the counterpart in the previous section. We shall first prove that the problem remains NP-hard.

Theorem 3. *The reclaimer scheduling problem with the restriction that any stockpile can be handled by only one reclaimer at a time is NP-hard.*

Proof. We reduce from the Partition Problem. Here, we are given a set of n items with size $a_1, \cdots, a_n$, and the total size is $\sum_{1 \le j \le n} a_j = A$. The question is whether there exists a set $B \subseteq \{1, \ldots, n\}$ which satisfies $\sum_{j \in B} a_j = A/2$.

We construct a scheduling instance of the problem as follows: two machines are used to schedule $n + 2$ jobs on three strips, in which n jobs are on the 2nd strip, one on the 1st and one on the 3rd, respectively.

$J_1^{(1)}$: $p_1^{(1)} = A/2,\ s_1^{(1)} = 1$.
$J_j^{(2)}$: $p_j^{(2)} = a_j,\ s_j^{(2)} = 1,$ for all $j \in \{1, \cdots, n\}$.
$J_1^{(3)}$: $p_1^{(3)} = A/2,\ s_1^{(3)} = 1$.

It is easy to verify that the answer to the partition problem is 'yes' if and only if the makespan of the corresponding scheduling instance is no more than $A + 1$. Thus, NP-hardness follows. □

4.1 2-approximation Algorithm for Multiple Machines

For the general case that the number m of machines is arbitrary, it is difficult to reduce the approximation ratio smaller than 2. Here we introduce a simple algorithm with approximation ratio of 2 by worst case analysis, and the time complexity of the algorithm is $O(n)$ in linear time. In the following subsection, we reduce the approximation ratio to 3/2 in the case of two machines. Later we provide numerical experiments to show the performance of the algorithm.

Denote by P_i the total processing time of the jobs on strip i, and denote by $t_i = s_{n_i}^{(i)}$ the maximal job location of the jobs on strip i. Fix an optimal solution and let C_i^* and C_i^{H2} be the load of M_i in the optimal solution and in the solution by Algorithm H2, respectively. The makespan generated by Algorithm H2 is denoted by C^{H2}.

Algorithm H2. *Assign all the jobs on strips 1 and 2 to M_1, and assign all the jobs from strip i $(2 < i \le m + 1)$ to M_{i-1}. Each machine travels from left to right to process the jobs in increasing order of job locations.*

Theorem 4. *Algorithm H2 gives a 2-approximation for the problem in which there are multiple machines and the machines are not allowed to work on the same stockpile simultaneously.*

Proof. In any optimal solution of this problem, all the jobs on strip 1 and 2 will be handled by M_1 and M_2. At least one of machines M_1 and M_2 needs to travel a distance of $\max\{t_1, t_2\}$. Therefore, we have

$$P_1 + P_2 + \max\{t_1, t_2\} \le C_1^* + C_2^* \le 2\mathrm{OPT}.$$

In the same way, all the jobs on strip i $(2 \le i \le m)$ can only be processed by M_{i-1} and M_i. So in any optimal solution, we have

$$P_i + t_i \le C_{i-1}^* + C_i^* \le 2\mathrm{OPT}.$$

The load of M_1 is $C_1^{H2} = P_1 + P_2 + \max\{t_1, t_2\} \le 2\mathrm{OPT}$, and the load of M_i is $C_i^{H2} = P_i + t_i \le 2\mathrm{OPT},\ (2 \le i \le m)$. Thus $C^{H2} \le 2\,\mathrm{OPT}$. □

It requires $O(n)$ time in total to assign n jobs to the corresponding machines according to Algorithm H2. The approximation ratio of Algorithm H2 is not better than 2 as can be seen from the following example with $m = 2$ machines and $n = 4$ jobs. Job J_1 is on strip 1, and J_2 is on strip 3. $p_1 = p_2 = s_1 = s_2 = \varepsilon$. J_3 and J_4 are on strip 2, and $p_3 = p,\ s_3 = \varepsilon, p_4 = \varepsilon,\ s_4 = p$. In the optimal solution of this instance, M_1 processes J_1, J_3 and M_2 processes J_2 first then travels forward to handle J_4. The optimal value is $p + 2\varepsilon$. The makespan generated by Algorithm H2 is $2p + \varepsilon$. When $p \to +\infty$, the approximation ratio is 2.

4.2 3/2-approximation Algorithm for Two Machines

In this subsection, a 3/2-approximation algorithm is presented when there are only two machines and three strips of jobs. The idea of the algorithm is to assign all jobs which belong to the same stockpile to the same machine and process these jobs consecutively. The routing of the machines will get simple under such assignments since it is optimal to move forward only.

Denote by $|B| = b$ the number of stockpiles on the middle (second) strip. The set of all the jobs on the same stockpile is defined as a *block*. Since all the jobs from the same block B_h have the same job location, say $s_h^{(2)}$, then $s_h^{(2)}$ is defined as the *block location* of the stockpile. If no ambiguity arises, the block location $s_h^{(2)}$ is simplified as s_h. The blocks $B_1, \ldots, B_b$ are sorted by the increasing order of the block locations. Since our algorithm does not distinguish between the different jobs in a single block we shall denote the total processing time of the jobs of block B_h simply by q_h. Without loss of generality, we assume that the last block on strip 1 is not further away than the last block on strip 3, i.e., $t_1 \leq t_3$.

Algorithm H3.

Step 1. *For $0 \leq i \leq b$:*

Step 1.1. *Assign all the jobs on strip 1 and the first i blocks on the middle strip to M_1.*

Step 1.2. *Assign all the jobs on strip 3 and the last $b-i$ blocks on the middle strip to M_2.*

Step 1.3. *Each machine travels from left to right to process the jobs that are assigned to it in the increasing order of job locations.*

Step 2. *Output the schedule with the smallest makespan among the $b+1$ schedules obtained by Step 1.*

Theorem 5. *Algorithm H3 gives 3/2-approximation for the problem in which there are only two machines and the machines cannot work on the same stockpile at the same time.*

Proof. Denote by P_1(or P_3) the total processing time of the jobs on strip 1 (or 3). Let C_1 and C_2 be the loads of machine M_1 and M_2 obtained by Algorithm H3, respectively. Assume that, by Algorithm H3, the solution is found by assigning the first i blocks to M_1.

The total processing time done by the two machines is $P_1 + P_3 + \sum_{h=1}^{b} q_h$ and at least one machine has to travel a distance of $\max\{t_1, t_2, t_3\} = \max\{t_2, t_3\}$ and the other has to travel to at least $\min\{t_1, t_2\} = t_1$. Hence, a lower bound on the optimal value is

$$LB = \frac{1}{2}(P_1 + P_3 + \sum_{h=1}^{b} q_h + \max\{t_2, t_3\} + t_1).$$

We shall distinguish between cases $i = 0$, $i = b$ and $1 \leq i \leq b-1$.

Case 1. $i = 0$. In this case, all jobs go to M_2. Thus we have $C_1 = P_1 + t_1$, and $C_2 = P_3 + \sum_{h=1}^{b} q_h + \max\{t_2, t_3\}$. According to the algorithm, it is not worse than the solution in which only B_1 goes to M_1 while keeping other assignments unchanged. Here the load of M_1 becomes $C_1' = P_1 + q_1 + \max\{t_1, s_1\}$. Because of the optimality of our choice for i, we must have $C_2 \leq C_1'$, and it follows that $C_2 \leq P_1 + q_1 + \max\{t_1, s_1\} = C_1 - t_1 + q_1 + \max\{t_1, s_1\}$.

So we have

$$C_2 - C_1 \leq -t_1 + q_1 + \max\{t_1, s_1\} \leq q_1 + s_1. \tag{1}$$

Further,

$$C_1 + C_2 = P_1 + P_3 + \sum_{h=1}^{b} q_h + t_1 + \max\{t_2, t_3\} = 2LB. \tag{2}$$

Combining (1) and (2), we get $2C_2 \leq 2LB + q_1 + s_1 \leq 3\text{OPT}$. Further, we may assume that $C_1 \leq C_2$ since otherwise the schedule produced by the algorithm is clearly optimal. Thus, also $C_2 \leq 3\text{OPT}/2$.

Case 2. $i = b$. In this case, all jobs are assigned on M_1. The loads of the machines are $C_1 = P_1 + \sum_{h=1}^{b} q_h + \max\{t_1, t_2\}$, and $C_2 = P_3 + t_3$. It is not worse than the assignment in which B_b is moved to M_2. Here, the load of machine M_2 is $C_2'' = P_3 + q_b + \max\{t_2, t_3\}$. It follows that $C_1 \leq C_2''$, so

$$C_1 \leq P_3 + q_b + \max\{t_2, t_3\} = C_2 + q_b + \max\{t_2, t_3\} - t_3.$$

By the inequality above, we get $C_1 - C_2 \leq q_b + \max\{t_2, t_3\} - t_3$. Further,

$$\begin{aligned} C_1 + C_2 =& P_1 + P_3 + \sum_{h=1}^{b} q_h + \max\{t_1, t_2\} + t_3 \\ =& 2LB - \max\{t_2, t_3\} - t_1 + \max\{t_1, t_2\} + t_3. \end{aligned}$$

Thus, $2C_1 \leq 2LB + q_b - t_1 + \max\{t_1, t_2\} \leq 3\text{OPT}$. We may assume that $C_2 \leq C_1$ since otherwise the schedule produced by the algorithm is clearly optimal. Thus, also $C_1 \leq 3\text{OPT}/2$.

Case 3. $1 \leq i \leq b - 1$. In this case, the first i blocks on the middle strip are assigned to M_1 and the rest to M_2, and the loads of machines generated by Algorithm H3 are $C_1 = P_1 + \sum_{h=1}^{i} q_h + \max\{t_1, s_i\}$, and $C_2 = P_3 + \sum_{h=i+1}^{b} q_h + \max\{t_2, t_3\}$. If we change the schedule by assigning block B_{i+1} to M_1 while keeping other assignments unchanged, then the completion time of machine M_2 will go down ($C_2' < C_2$) while the completion time of M_1 changes to $C_1' = P_1 + \sum_{h=1}^{i+1} q_h + \max\{t_1, s_{i+1}\}$. By the optimality of our choice for i, we must have $C_2 \leq C_1'$, which implies

$$\sum_{h=1}^{i} q_h - \sum_{h=i+1}^{b} q_h \geq P_3 - P_1 + \max\{t_2, t_3\} - \max\{t_1, s_{i+1}\} - q_{i+1}.$$

Subtracting $2LB$ from both sides gives

$$\sum_{h=i+1}^{b} q_h \leq LB - P_3 - \max\{t_2, t_3\} - \frac{1}{2}t_1 + \frac{1}{2}\max\{t_1, s_{i+1}\} + \frac{1}{2}q_{i+1}.$$

Hence,

$$\begin{aligned} C_2 =& P_3 + \sum_{h=i+1}^{b} q_h + \max\{t_2, t_3\} \\ \leq& P_3 + LB - P_3 - \max\{t_2, t_3\} - \frac{1}{2}t_1 + \frac{1}{2}\max\{t_1, s_{i+1}\} + \frac{1}{2}q_{i+1} + \max\{t_2, t_3\} \\ =& LB - \frac{1}{2}t_1 + \frac{1}{2}\max\{t_1, s_{i+1}\} + \frac{1}{2}q_{i+1} \\ \leq& \frac{3}{2}\text{OPT}. \end{aligned}$$

Similarly, if block B_i is moved to M_2 while keeping other assignments unchanged, then the completion time of machine M_1 will go down while the completion time of M_2 changes to

$C_2'' = P_3 + \sum_{h=i}^{b} q_h + \max\{t_2, t_3\}$. By the optimality of our choice for i, we have $C_1 \le C_2''$, which implies

$$\sum_{h=1}^{i} q_h - \sum_{h=i+1}^{b} q_h \le P_3 - P_1 + \max\{t_2, t_3\} - \max\{t_1, s_i\} + q_i.$$

Adding $2LB$ to both sides gives

$$\sum_{h=1}^{i} q_h \le LB - P_1 - \frac{1}{2}\max\{t_1, s_i\} - \frac{1}{2}t_1 + \frac{1}{2}q_i.$$

Therefore,

$$\begin{aligned} C_1 =& P_1 + \sum_{h=1}^{i} q_h + \max\{t_1, s_i\} \\ \le& P_1 + LB - P_1 - \frac{1}{2}\max\{t_1, s_i\} - \frac{1}{2}t_1 + \frac{1}{2}q_i + \max\{t_1, s_i\} \\ =& LB + \frac{1}{2}\max\{t_1, s_i\} - \frac{1}{2}t_1 + \frac{1}{2}q_i \\ \le& \frac{3}{2}\mathrm{OPT}. \end{aligned}$$

We proved that the Algorithm H3 gives a 3/2-approximation for the problem. It requires $O(n \log n)$ time to sort the three strips of jobs in the increasing order of job locations. The algorithm needs to check at most $O(n)$ different assignments and pick the best one and each assignment requires at most $O(n)$ time to calculate the objective value. So, the overall time complexity of Algorithm H3 is $O(n^2)$. □

The approximation ratio of Algorithm H3 is not better than 3/2 as can be seen from the example above Theorem 3. In instance $\widehat{I}$, the makespan obtained by Algorithm H3 is $3p$, but the optimal value is $2p + \varepsilon$. Thus, when $p \to +\infty$, the approximation ratio is 3/2.

5 Numerical Experiments

In this section, we give numerical simulations to evaluate the performance of Algorithm H2 and H3. The computational results are shown by comparing the values of the approximation algorithms with lower bounds on the optimal value. As a result, the average error of H2 and H3 compared with the lower bounds is 0.4593 and 0.0242, respectively, much better than our worst case analysis.

The lower bound is derived from the following integer programming. Note that any solution for the problem in which the machines are not allowed to work on the same stockpile (in Section 4) is also a solution for the problem in which the machines are allowed to work on the same stockpile (in Section 3). Therefore, the latter problem is a relaxation of the former. We use an integer programming (IP) to find optimal solutions for the later problem and use it as a lower bound.

$$\begin{aligned} &\min C_{\max} \\ &\text{s.t. } l_i + \sum_{1 \le j \le n_i} (1 - x_j^{(i)}) p_j^{(i)} + \sum_{1 \le j \le n_{i+1}} x_j^{(i+1)} p_j^{(i+1)} \le C_{\max}, \qquad \text{for all } 1 \le i \le m, \qquad (3) \\ &\qquad l_i \ge (1 - x_j^{(i)}) s_j^{(i)}, \qquad \text{for all } \ 1 \le i \le m, \ \ 1 \le j \le n_i, \qquad (4) \end{aligned}$$

$$l_i \geq x_j^{(i+1)} s_j^{(i+1)}, \qquad \text{for all } \ 1 \leq i \leq m, \ \ 1 \leq j \leq n_{i+1}, \tag{5}$$

$$x_j^{(1)} = 0, \qquad \text{for all } \ 1 \leq j \leq n_1, \tag{6}$$

$$x_j^{(m+1)} = 1, \qquad \text{for all } \ 1 \leq j \leq n_{m+1}, \tag{7}$$

$$x_j^{(i)} \in \{0, 1\}, \qquad \text{for all } \ J_j^{(i)}. \tag{8}$$

The boolean variable $x_j^{(i)} = 1$ indicates that job $J_j^{(i)}$ is assigned to machine M_{i-1}. If $x_j^{(i)} = 0$ then it is assigned to M_i. Constraint (3) ensures that the load of each machine is no more than the objective value $C_{\max}$. Here, l_i is the location of the right-most job done by M_i. Equations (6) and (7) indicate that the jobs on the first (last) strip can only be processed by the first (last) machine. Denote by LB_{IP} the lower bound obtained from solving (IP).

Algorithm H2 and H3 were implemented in Python 3.4 and the integer programming model was coded in Gurobi 6.0.3. All the programmes run on a desktop PC with a 4.5 GHz Core i7 CPU and 16 GB RAM.

To analyze the effect of different number of jobs and machines by the algorithms, we generate test instances under different conditions. Each condition is evaluated by generating 100 test instances, respectively. The evaluation of each condition consists of the mean relative error and the running time of the condition. The mean relative error is the average ratio of $(C_{\max}^H - lb)$ to lb, where $C_{\max}^H$ is the makespan generated by the algorithm and lb is a lower bound.

Given the number $|J|$ of jobs of a test instance, the number of jobs per strip is chosen uniformly at random, i.e. each of the $|J|$ jobs is assigned with the probability of $\frac{1}{m+1}$ to each of the strips. The processing time of each job is generated uniformly at random from $[1, 100]$. Both the number of stockpiles per strip and the location of each stockpile are also generated randomly. After the number of stockpiles per strip is fixed, the location of each stockpile is chosen uniformly at random from $[1, 300]$. Then the job is assigned to a stockpile with the probability of $\frac{1}{x_i}$, where x_i is the number of stockpiles on i-th strip. Thus, each of the job on the strip is allocated uniformly at random to one of the stockpiles on the strip. If there is no job assigned to the stockpile after the allocation, then remove that stockpile from the instance.

Table 1. The Performance of Algorithm H2

	Mean relative error			Mean running time(s)		
n	$m = 2$	$m = 5$	$m = 10$	$m = 2$	$m = 5$	$m = 10$
20	0.2842	0.4142	0.3766	0.0016	0.0027	0.0028
50	0.3219	0.4976	0.4254	0.0020	0.0186	0.0097
100	0.3160	0.5480	0.4976	0.0021	0.1515	0.1294
200	0.3324	0.6014	0.6192	0.0021	0.3367	3.9359
500	0.3280	0.6523	0.6742	0.0037	1.5746	19.0245

Table 1 shows the mean relative error of Algorithm H2 with different number of machines and jobs. We can see that the performance of Algorithm H2 is better than the worst case guarantees. Here the lower bound is LB_{IP}. The mean relative error reaches the lowest value of 0.2842 when there are 2 machines and 20 jobs, and it gets highest as 0.6742 in the condition of 10 machines and 500 jobs. The mean relative error will further decrease if better lower bounds can be found. Table 1 implies the number of jobs and machines have an impact on the performance of the algorithm, which performs better when the size of the instance is small.

If there are only two machines and three strips of jobs, two obvious lower bounds are $LB_1 = \max\{P_1 + t_1, P_3 + t_3\}$ and $LB_2 = \max\limits_{h \in \{1, \cdots, b\}} \{q_h + s_h\}$. Thus, we obtain a better lower bound by taking $\overline{LB} = \max\{LB, LB_1, LB_2\}$.

Table 2. The performance of H3 and the comparison between H2 and H3

	Mean relative error			Running time(s)	
n	LB_{IP}	$\overline{LB}$	H2 ($m = 2$)	H3	H2 ($m = 2$)
20	0.0268	0.0292	0.2842	0.0016	0.0016
50	0.0267	0.0295	0.3219	0.0022	0.0020
100	0.0234	0.0232	0.3160	0.0020	0.0021
200	0.0226	0.0225	0.3324	0.0030	0.0021
500	0.0191	0.0191	0.3280	0.0031	0.0037

We evaluate the performance of Algorithm H3 by two lower bounds, $\overline{LB}$ and LB_{IP}. Table 2 shows that the computational results of Algorithm H3 and the comparison with H2 under the condition that $m = 2$. We can see from the table that the algorithm is very fast, which solves the instances of 500 jobs within 0.003 seconds. In comparison with H2 when $m = 2$, the performance of H3 is always better. In particular, Algorithm H3 performs better even when the size of the instance gets bigger, and the ratio reaches 0.0191 with 500 jobs, much lower than the worst case guarantees. However, Algorithm H2 performs worse with the size of the instance increasing, which grows from 0.2842 to 0.3280 with the number of jobs increasing. Therefore, when there are only two machines, Algorithm H3 outperforms H2 without taking more running times.

6 Conclusion

We discuss the reclaimer scheduling problem of the reclaiming system from the perspective of combinatorial optimization. Two variants of the problem are considered in this paper: the one where two machines can work on the same stockpile at the same time, and the variant in which only one machine can work on a stockpile at a time. Both problems are NP-hard. The first one has an FPTAS. The other variant appears much harder and we design an approximation algorithm with the performance ratio of 2 for the general case, and an approximation algorithm with the ratio of 3/2 when there are two identical machines. The numerical simulations show that the average error of Algorithm H2 and H3 compared with the lower bounds is 0.4593 and 0.0242, respectively. These errors on the test instances are much better than the worse case ratio of 1 and 0.5 which follows from our theoretical analysis.

References

[1] Alon, N., Azar, Y., Woeginger, G., Yadid, T. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1): 55–66 (1998)

[2] Angelelli, E., Kalinowski, T., Kapoor, R., Savelsbergh, M. A reclaimer scheduling problem arising in coal stockyard management. *Journal of Scheduling*, 19(5): 563–582 (2016)

[3] Ebenlendr, T., Krčál, M., Sgall, J. Graph balancing: a special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1): 62–80 (2014)

[4] Daganzo, C.F. The crane scheduling problem. *Transportation Research Part B*, 23(3): 159–175 (1989)

[5] Garey, M., Johnson, D. "strong" np-completeness results: motivation, examples, and implications. *Journal of the ACM*, 25(3): 499–508 (1978)

[6] Garey, M., Johnson, D. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco, 1979

[7] Hu, D., Yao, Z. Stacker-reclaimer scheduling in a dry bulk terminal. *International Journal of Computer Integrated Manufacturing*, 25(11): 1047–1058 (2012)

[8] Kim, K., Kim, K. A routing algorithm for a single straddle carrier to load export containers onto a containership. *International Journal of Production Economics*, 59(1): 425–433 (1999)

[9] Kim, K., Kim, K. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. *Naval Research Logistics*, 50(5): 498–514 (2003)

[10] Lee, K., Bae, H., Hong, S. Approximation of optimal moving paths of huge robot reclaimer with a 3D range finder. In: Proceedings of Computational Science and Its Applications, ed. by Marina Gavrilova et al., Springer, 2006, 151–160

[11] Lee, K., Leung, J., Pinedo, M. A note on graph balancing problems with restrictions. *Information Processing Letters*, 110(1): 24–29 (2009)

[12] Lu, T. Bucket wheel reclaimer modeling as a robotic arm. In: Proceedings of IEEE International Conference on Robotics and Biomimetics, China, Guilin, 2009, 263-268

[13] Lu, T., Myo, M. Optimization of reclaiming voxels for quality grade target with reclaimer minimum movement. In: Proceedings of 11th International Conference on Control Automation Robotics and Vision, Singapore, 2010, 341–345

[14] Menezes, G.C., Mateus, G.R., Ravetti, M.G. A hierarchical approach to solve a production planning and scheduling problem in bulk cargo terminal. *Computers and Industrial Engineering*, 97: 1–14 (2016)

[15] Menezes, G.C., Mateusm, G.R., Ravetti, M.G. A branch and price algorithm to solve the integrated production planning and scheduling in bulk ports. *European Journal of Operational Research*, 258(3): 926–937 (2017)

[16] National Bureau of Statistics of China (NBSC). China Statistical Yearbook 2016. China Statistical Press, Beijing, 2016

[17] Ng, W.C. Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, 164(1): 64–78 (2005)

[18] Ng, W.C., Mak, K.L. An effective heuristic for scheduling a yard crane to handle jobs with different ready times. *Engineering Optimization*, 37(8): 867–877 (2005)

[19] Robenek, T., Umang, N., Bierlaire, M., Ropke, S. A branch-and-price algorithm to solve the integrated berth allocation and yard assignment problem in bulk ports. *European Journal of Operational Research*, 235(2): 399–411 (2014)

[20] Sun, D., Tang, L. Benders approach for the raw material transportation scheduling problem in steel industry. In: Proceedings of 10th International Conference on Control and Automation, China, Hangzhou, 2013, 481–484

[21] Zhang, C., Wan, Y., Liu, J., Linn, R. Dynamic crane deployment in container storage yards. *Transportation Research Part B*, 36(6): 537–555 (2002)

[22] Zhao, S., Lu, T., Koch, B., Hurdsman, A. Stockpile modelling using mobile laser scanner for quality grade control in stockpile management. In: Proceedings of 12th IEEE International Conference on Control, Automation, Robotics and Vision, China, Guangzhou, 2012, 811–816

[23] Zhao, S., Lu, T., Koch, B., Hurdsman, A. A simulation study of sensor data fusion using UKF for bucket wheel reclaimer localization. In: Proceedings of 8th International Conference on Automation Science and Engineering, Korea, Seoul, 2012, 1192–1197